**Input:**

**Sender Side:**

import java.util.Scanner;

public class BinarySegmentAdder {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        // Step 1: Get the number of bits

        System.out.print("Enter the number of bits of data: ");

        int numBits = scanner.nextInt();

        scanner.nextLine(); // Consume newline

        String binaryData;

        while (true) {

            // Step 2: Get the binary data

            System.out.print("Enter the binary data (" + numBits + " bits): ");

            binaryData = scanner.nextLine();

            if (binaryData.matches("[01]{" + numBits + "}")) {

                break;

            } else {

                System.out.println("Invalid input! Please enter exactly " + numBits + " bits in binary form (0s and 1s only).");

            }

        }

        // Step 3: Get the number of segments

        int segments;

        while (true) {

            System.out.print("Enter the number of segments to divide the data into: ");

            segments = scanner.nextInt();

            // Ensure valid segment count

            if (numBits % segments == 0) {

                break;

            } else {

                System.out.println("Invalid input! The data cannot be evenly divided into " + segments + " segments.");

            }

        }

        // Step 4: Divide the binary data into segments

        int segmentSize = numBits / segments;

        String[] segmentArray = new String[segments];

        System.out.println("\nDivided Segments:");

        for (int i = 0; i < segments; i++) {

            segmentArray[i] = binaryData.substring(i \* segmentSize, (i + 1) \* segmentSize);

            System.out.println(segmentArray[i]);

        }

        // Step 5: Perform binary addition with carry discard rule

        String result = segmentArray[0]; // Start with first segment

        for (int i = 1; i < segments; i++) {

            result = addBinaryWithCarryDiscard(result, segmentArray[i]);

            System.out.println("Result after iteration "+i+": "+result);

        }

        // Step 6: Perform one's complement

        String onesComplement = onesComplement(result);

        // Step 7: Generate the final codeword (data + one's complement checksum)

        String finalCodeword = binaryData + onesComplement;

        // Display results

        System.out.println("\nFinal sum after binary addition: " + result);

        System.out.println("Checksum obtained is: " + onesComplement);

        System.out.println("\nFinal Codeword (Data + Checksum): " + finalCodeword);

        scanner.close();

    }

    // Function to perform binary addition with carry discard rule

    public static String addBinaryWithCarryDiscard(String bin1, String bin2) {

        int num1 = Integer.parseInt(bin1, 2);

        int num2 = Integer.parseInt(bin2, 2);

        int sum = num1 + num2; // Perform binary addition

        String sumBinary = Integer.toBinaryString(sum); // Convert to binary string

        // If the sum exceeds segment size, discard the carry

        if (sumBinary.length() > bin1.length()) {

            sumBinary = sumBinary.substring(1); // Remove the carry

            // Convert back to integer, add 1 to the least significant bit

            int adjustedSum = Integer.parseInt(sumBinary, 2) + 1;

            sumBinary = String.format("%" + bin1.length() + "s", Integer.toBinaryString(adjustedSum)).replace(' ', '0'); // Ensure correct format

        } else {

            sumBinary = String.format("%" + bin1.length() + "s", sumBinary).replace(' ', '0'); // Ensure correct format

        }

        return sumBinary;

    }

    // Function to compute one's complement

    public static String onesComplement(String binary) {

        StringBuilder complement = new StringBuilder();

        for (char bit : binary.toCharArray()) {

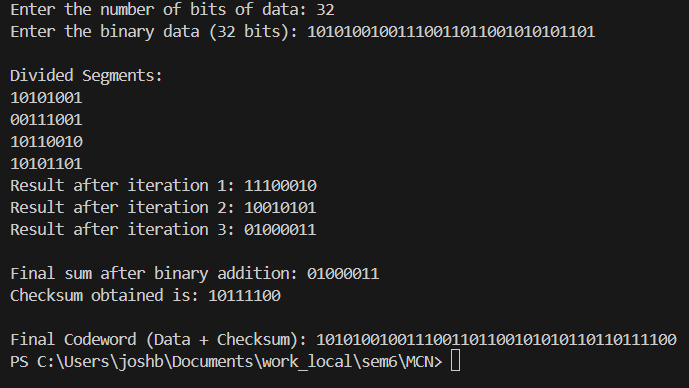
            complement.append(bit == '0' ? '1' : '0'); // Flip bits

        }

        return complement.toString();

    }

}

**Output:** ****

**Input:**

**Receiver side:**

import java.util.Scanner;

public class BinaryChecksumReceiver {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Get the number of bits (Data + Checksum)

System.out.print("Enter the total number of bits received (Data + Checksum): ");

int numBits = scanner.nextInt();

scanner.nextLine();

String receivedData;

while (true) {

// Get the binary data

System.out.print("Enter the received binary data (" + numBits + " bits): ");

receivedData = scanner.nextLine();

// Validate that the input is binary and matches the required length

if (receivedData.matches("[01]{" + numBits + "}")) {

break;

} else {

System.out.println("Invalid input! Please enter exactly " + numBits + " bits in binary form (0s and 1s only).");

}

}

// Get the number of segments

int segments;

while (true) {

System.out.print("Enter the number of segments: ");

segments = scanner.nextInt();

// Ensure valid segment count

if (numBits % segments == 0) {

break;

} else {

System.out.println("Invalid input! The data cannot be evenly divided into " + segments + " segments.");

}

}

// Step 4: Divide received data into segments

int segmentSize = numBits / segments;

String[] receivedSegments = new String[segments];

System.out.println("\nReceived Segments:");

for (int i = 0; i < segments; i++) {

receivedSegments[i] = receivedData.substring(i \* segmentSize, (i + 1) \* segmentSize);

System.out.println(receivedSegments[i]);

}

// Step 5: Perform binary addition

String result = receivedSegments[0]; // Start with first segment

for (int i = 1; i < segments; i++) {

result = addBinaryWithCarryDiscard(result, receivedSegments[i]);

System.out.println("Result after iteration "+i+": "+result);

}

// Step 6: Compute one's complement of the sum

String finalComplement = onesComplement(result);

// Display receiver side results

System.out.println("\nSum after adding all segments (including checksum): " + result);

System.out.println("Checksum obtained is: " + finalComplement);

// Step 7: Dynamic Error Check

String zeroString = "0".repeat(segmentSize); // Generate dynamic zero string of segment size

if (finalComplement.equals(zeroString)) {

System.out.println("\nNo error detected. Data transmission successful!");

} else {

System.out.println("\nError detected in transmission!");

}

// Step 8: Extract and display the transmitted data (without the checksum)

String transmittedData = receivedData.substring(0, numBits - segmentSize);

System.out.println("\nTransmitted Data (without checksum): " + transmittedData);

scanner.close();

}

// Function to perform binary addition with carry discard rule

public static String addBinaryWithCarryDiscard(String bin1, String bin2) {

int num1 = Integer.parseInt(bin1, 2);

int num2 = Integer.parseInt(bin2, 2);

int sum = num1 + num2; // Perform binary addition

String sumBinary = Integer.toBinaryString(sum); // Convert to binary string

// If the sum exceeds segment size, discard the carry

if (sumBinary.length() > bin1.length()) {

sumBinary = sumBinary.substring(1); // Remove the carry

// Convert back to integer, add 1 to the least significant bit

int adjustedSum = Integer.parseInt(sumBinary, 2) + 1;

sumBinary = String.format("%" + bin1.length() + "s", Integer.toBinaryString(adjustedSum)).replace(' ', '0'); // Ensure correct format

} else {

sumBinary = String.format("%" + bin1.length() + "s", sumBinary).replace(' ', '0'); // Ensure correct format

}

return sumBinary;

}

// Function to compute one's complement

public static String onesComplement(String binary) {

StringBuilder complement = new StringBuilder();

for (char bit : binary.toCharArray()) {

complement.append(bit == '0' ? '1' : '0'); // Flip bits

}

return complement.toString();

}

}

**Output:**

